

Internet Engineering Task Force
INTERNET DRAFT
File: [draft-ietf-tsvwg-dsack-use-00.txt](#)

Ethan Blanton
Purdue University
Mark Allman
BBN/NASA GRC
June, 2003
Expires: December, 2003

Using TCP DSACKs and SCTP Duplicate TSNs to Detect Spurious Retransmissions

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of \[RFC2026\]](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Abstract

TCP and SCTP provide notification of duplicate segment receipt through DSACK and Duplicate TSN notification, respectively. This document presents a conservative method of using this information to identify unnecessary retransmissions for various applications.

Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

1 Introduction

TCP [[RFC793](#)] and SCTP [[RFC2960](#)] provide notification of duplicate segment receipt through duplicate selective acknowledgment (DSACK) [[RFC2883](#)] and Duplicate TSN notifications, respectively. Using this information, a TCP or SCTP sender can generally determine when a retransmission was sent in error. This document presents two methods for using duplicate notifications. The first method is

simple and can be used for accounting applications. The second method is a conservative algorithm to disambiguate unnecessary retransmissions from loss events for the purpose of undoing

Expires: December 2003

[Page 1]

unnecessary congestion control changes.

This document is intended to outline reasonable and safe algorithms for detecting spurious retransmissions and discuss some of the considerations involved. It is not intended to describe the only possible method for achieving the goal, although the guidelines in this document should be taken into consideration when designing alternate algorithms. Additionally, this document does not outline what a TCP or SCTP sender may do after a spurious retransmission is detected. A number of proposals have been developed (e.g., [[RFC3522](#)], [[SK03](#)]), but it is not yet clear which of these proposals are appropriate. In addition, they all rely on detecting spurious retransmits and so can share the algorithm specified in this document.

2 Counting Duplicate Notifications

For certain applications a straight count of duplicate notifications will suffice. For instance, if a stack simply wants to know (for some reason) the number of spuriously retransmitted segments, counting all duplicate notifications for retransmitted segments should work well. Another application of this strategy is to monitor and adapt transport algorithms so that the transport is not sending large amounts of spurious data into the network. For instance, monitoring duplicate notifications could be used by the Early Retransmit [[AAAB03](#)] algorithm to determine whether fast retransmitting [[RFC2581](#)] segments with a lower than normal duplicate ACK threshold is working or if segment reordering is causing spurious retransmits.

3 Congestion/Duplicate Disambiguation Algorithm

When the purpose of detecting spurious retransmissions is to ``undo'' unnecessary changes made to the congestion control state, as suggested in [[RFC2883](#)], the data sender ideally needs to determine:

- (a) That spurious retransmissions in a particular window of data do not mask real segment loss (congestion).

For example, assume segments N and N+1 are retransmitted even though only segment N was dropped by the network (thus, segment N+1 was needlessly retransmitted). When the sender receives the notification that segment N+1 arrived more than once it can conclude that segment N+1 was needlessly resent. However, it cannot conclude that it is appropriate to revert the congestion control state because the window of data contained at least one valid congestion indication (i.e., segment N was lost).

(b) That network duplication is not the cause of the duplicate notification.

Determining whether a duplicate notification is caused by

Expires: December 2003

[Page 2]

network duplication or a spurious retransmit is a nearly impossible task in theory. Since [[Pax97](#)] shows that packet duplication by the network is rare the algorithm in this section simply ceases to function when network duplication is detected (by receiving a duplication notification for a segment that was not retransmitted by the sender).

The algorithm specified below gives reasonable protection against both of these cases.

We assume the TCP sender has a data structure to hold selective acknowledgment information (e.g., as outlined in [[RFC3517](#)]). The following steps MUST be taken upon the receipt of each DSACK or duplicate TSN notification:

- (A) Check the corresponding sequence range or TSN to determine whether the segment has been retransmitted.
 - (A.1) If the segment was retransmitted, mark it as a duplicate.
 - (A.2) If the segment was not retransmitted the incoming DSACK indicates that the network duplicated the segment in question. Processing of this DSACK MUST be terminated. In addition, the algorithm specified in this document MUST NOT be used for the remainder of the connection, as future DSACK reports may be indicating network duplication rather than unnecessary retransmission. Note that some techniques to further disambiguate network duplication from unnecessary retransmission (e.g., the TCP timestamp option [[RFC1323](#)]) may be used to refine the algorithm in this document further. Using such a technique in conjunction with an algorithm similar to the one presented herein may allow for the continued use of the algorithm in the face of duplicated segments. We do not delve into such an algorithm in this document due the current rarity of network duplication. However, future work should include tackling this problem.
- (B) Check all retransmitted segments in the previous window of data.
 - (B.1) If all segments or chunks marked as retransmitted have also been marked as duplicate, we conclude that all retransmissions in the previous window of data were spurious and no loss occurred.
 - (B.2) If any segment or chunk is still marked as retransmitted but not marked as duplicate, there are outstanding retransmissions that could indicate loss within this window of data. We can make no conclusions based on this

particular DSACK/duplicate TSN notification.

In addition to keeping the state mentioned in [[RFC3517](#)] (for TCP) and [[RFC2960](#)] (for SCTP), an implementation of this algorithm must track all sequence numbers or TSNs that have been acknowledged as

Expires: December 2003

[Page 3]

duplicates.

4 Related Work

In addition to the mechanism for detecting spurious retransmits outlined in this document, several other proposals for finding needless retransmits have been developed.

[BA02] uses the algorithm outlined in this document as the basis for investigating several methods to make TCP more robust to reordered packets.

The Eifel detection algorithm [RFC3522] uses the TCP timestamp option [RFC1323] to determine whether the ACK for a given retransmit is for the original transmission or a retransmission. More generally, [LK00] outlines the benefits of detecting spurious retransmits and reverting from needless congestion control changes using the timestamp-based scheme or a mechanism that uses a "retransmit bit" to flag retransmits (and ACKs of retransmits). The Eifel detection algorithm can detect spurious retransmits more rapidly than a DSACK-based scheme. However, the tradeoff is that the overhead of the 12-byte timestamp option must be incurred in every packet transmitted for Eifel to function.

The F-RTT scheme [SK03] slightly alters TCP's sending pattern immediately following a retransmission timeout and then observes the pattern of the returning ACKs. This pattern can indicate whether the retransmitted segment was needed. The advantage of F-RTT is that the algorithm only needs to be implemented on the sender side of the TCP connection and that nothing extra needs to cross the network (e.g., DSACKs, timestamps, special flags, etc.). The downside is that the algorithm is a heuristic that can be confused by network pathologies (e.g., duplication or reordering of key packets). Finally, note that F-RTT only works for spurious retransmits triggered by the transport's retransmission timer.

Finally, [AP99] briefly investigates using the time between retransmitting a segment via the retransmission timeout and the arrival of the next ACK as an indicator of whether the retransmit was needed. The scheme compares this time delta with a fraction (f) of the minimum RTT observed thus far on the connection. If the time delta is less than $f \cdot \text{minRTT}$ then the retransmit is labeled spurious. When $f=1/2$ the algorithm identifies roughly 59% of the needless retransmission timeouts and identifies needed retransmits only 2.5% of the time. As with F-RTT, this scheme only detects spurious retransmits sent by the transport's retransmission timer.

5 Security Considerations

It is possible for the receiver to falsely indicate spurious retransmissions in the case of actual loss, potentially causing a TCP or SCTP sender to inaccurately conclude that no loss took place (and possibly cause inappropriate changes to the senders congestion control state). Consider the following scenario:

Expires: December 2003

[Page 4]

A receiver watches every segment or chunk that arrives and acknowledges any segment that arrives out of order by more than some threshold amount as a duplicate, assuming that it is a retransmission. A sender using the above algorithm will assume that the retransmission was spurious.

The ECN nonce sum proposal [[SWE02](#)] would help mitigate the ability of the receiver to hide real losses from the sender. In the common case of receiving an original transmission and a spurious retransmit a TCP receiver will have received the nonce from the original transmission and therefore can "prove" to the sender that the duplication notification is valid.

References

- [AAAB03] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton. Early Retransmit for TCP. Internet-Draft [draft-allman-tcp-early-rexmt-00.txt](#), February 2003. Work in progress.
- [AP99] Allman, M. and V. Paxson, "On Estimating End-to-End Network Path Properties", SIGCOMM 99.
- [BA02] E. Blanton, M. Allman. On Making TCP More Robust to Packet Reordering. ACM Computer Communication Review, 32(1), January 2002.
- [LK00] R. Ludwig, R. H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. ACM Computer Communication Review, 30(1), January 2000.
- [Pax97] V. Paxson. End-to-End Internet Packet Dynamics. In ACM SIGCOMM, September 1997.
- [RFC793] Jon Postel. Transmission Control Protocol. Std 7, [RFC 793](#). September 1981.
- [RFC1323] Van Jacobson, Robert Braden, David Borman. TCP Extensions for High Performance. [RFC 1323](#). May 1992.
- [RFC2883] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. [RFC 2883](#), July 2000.
- [RFC2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. Stream Control Transmission Protocol. October 2000.
- [RFC3517] Ethan Blanton, Mark Allman, Kevin Fall, Lili Wang. A

Conservative Selective Acknowledgment (SACK)-based Loss Recovery
Algorithm for TCP, April 2003. [RFC 3517](#).

[RFC3522] R. Ludwig, M. Meyer. The Eifel Detection Algorithm for

Expires: December 2003

[Page 5]

TCP, April 2003. [RFC 3522](#).

[SK03] P. Sarolahti, M. Kojo. F-RTT: An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and SCTP. Internet-Draft [draft-sarolahti-tsvwg-tcp-frto-04.txt](#), June 2003. Work in progress.

[SWE02] N. Spring, D. Wetherall, D. Ely. Robust ECN Signaling with Nonces. Internet-Draft [draft-ietf-tsvwg-tcp-nonce-04.txt](#), October 2002. Work in progress.

Authors' Addresses:

Ethan Blanton
Purdue University Computer Sciences
1398 Computer Science Building
West Lafayette, IN 47907
ebanton@cs.purdue.edu

Mark Allman
BBN Technologies/NASA Glenn Research Center
Lewis Field
21000 Brookpark Rd. MS 54-5
Cleveland, OH 44135
Phone: 216-433-6586
Fax: 216-433-8705
mallman@bbn.com
<http://roland.grc.nasa.gov/~mallman>

Expires: December 2003

[Page 6]